A Neural Algorithm for Artistic Style

E4040.2017Fall.KMNN.report
Nachiket Paranjape nmp2139, Ketan Mehta kmm2304, Mohneesh Patel mp3542
Columbia University

Abstract

We introduce an artificial system based on a Deep Neural Network that creates artistic images of high perceptual quality. The system uses neural representations to separate and recombine content and style of arbitrary images, providing a neural algorithm for the creation of artistic images. The algorithm captures the style information within different layers of the network, then construct the image that matches the style representation of the given input image. The main technical challenge we faced was to train network which is extremely effective at object recognition - as a basis for trying to extract content and style representations from images. We overcame this challenge by using VGG16 network which is known for object recognition with 7% error. We implemented VGG16 and Alexnet for artistic style output compared to VGG19 implemented in the original paper. We could achieved our goal successfully by combining different styles with different content images that with the shallower network.

1. Introduction

Painting is a popular form of art. For hundreds of years, people have been attracted by the art of painting with the advent of many fantastic artworks, e.g., Vincent van Gogh's "The Starry Night". In the past, redrawing an image in a particular style manually required a well-trained artist and lots of time.

Since the mid-1990s, the art theories behind the fantastic artworks have been attracting the attention of not only the artists but many computer science researchers. There are plenty of studies exploring how to automatically turn images into synthetic artworks such that everyone can be an artist.

Furthermore, with the advent of new platforms such as Snapchat and Instagram that create accessible avenues for art technology to the public, the problem has become all the more relevant to the scientific community. With the motivation of reducing computational learning overhead as well as run time

With the advent of CNNs, we are able to tackle the problem in a more sophisticated manner. The pioneering work of Gatys et al. has attracted wide attentions both academically and industrially. In academia, lots of follow-up studies were proposed to either improve or extend this innovative algorithm and before long, these technologies were applied to many successful industrial applications. However, to the best of our knowledge, there are no comprehensive survey summarizing and discussing recent significant advances and challenges within this new field of Neural Style Transfer.

In that paper Gatys discussed, about CNN for style transfer. When Convolutional Neural Networks are trained on object recognition, they develop a representation of the image that makes object information increasingly explicit along the processing hierarchy [7].

We can directly visualise the information each layer contains about the input image by reconstructing the image only from the feature maps in that layer[4]:higher layers in the network capture the high level content in terms of objects and their arrangement in the input image but do not constrain the exact pixel values of the reconstruction while the reconstructions from the lower layers simply reproduce the exact pixel values of the original image. Therefore, we refer to the feature responses in higher layers of the network as the content representation.

The style can be thought as something independent of content, something we are left with if we let the content off. L. Gatys suggests to dismiss spatial information by computing correlations between the feature maps.

Our project aims to explore the CNN structures of Style Transfer algorithms proposed by the Gatys paper [2]. Our contributions are threefold. First, we implemented VGG16 model as discussed in the Gatys paper. Second, we also implemented Alexnet and compared the results with VGG16. Third, we tried SGD and Adam optimizer and compare the outputs of both the methods.

The rest of this paper is organized as follows. Section 2 covers the summary of the original paper [6]. It talks about the methods and explains these methods in detail. It also talks about key results discussed in the original paper [6]. Section 3 and Section 4 discusses the methodology and implementation of our approach. Then Section 5 provides results and comparison of our results with respect to original paper results. Section 6 concludes the paper.

2. Summary of the Original Paper

2.1 Methodology of the Original Paper

Inspired by the Convolutional Neural Network (CNN), Gatys et al. First explored how to use CNN to reproduce well-known painting styles on natural images. The images they obtained from CNN indicated that the representation of image content and style was divisible. Based on this finding, Gatys et al. [3] proposed a neural style transfer algorithm to restructure the style of a given photo content and well-known artwork. Their proposed algorithm succeeded in producing a fantastic style image with a specific artistic appearance.

Given a content image C and style image S the paper aims to generate an image X with content from C and style from S. For loss function it is desired to make an intermediate representation of X close to C i.e Loss for content is minimum || Fx - Fc||, where Fx and Fc are feature representations of image X and C respectively and and $F \in C \times W \times H$. As shown in [4,], [5] we know that the input image is easily invertible from the features representation of intermediate layers.

This paper proposes to eliminate spatial information by calculating the correlation between feature maps of expected values within the spatial range of the input image. These features correlations are given by Gram Matrix $G \in C \times C$. [6]The loss function defined is the following : $Lstyle = \min_x || G(F\,x) - G(F\,c)||$.

In order to generate an image that blends the content of the photo with the painting style, the paper collectively displays the content of the photo in one layer of the network as the distance from the white noise image and the number of layers in the CNN. So the loss function defined is the following: $L = \alpha Lcontent + \beta Lstyle$.

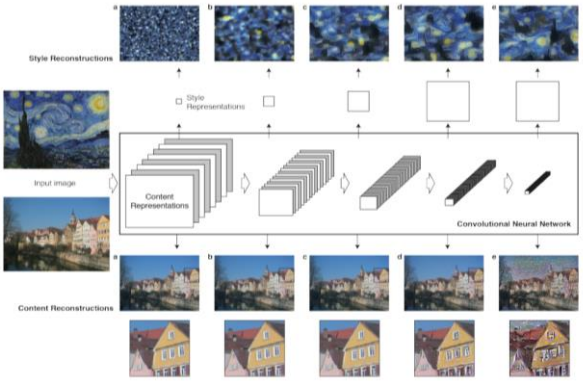

Figure 1: Convolutional Neural Network (CNN)The given input image is represented as a collection filter images for each processing phase of CNN. The number of different filters As processing levels increase, the size of the filtered image is reduced The downsampling mechanism (e.g. max pooling) leads to a reduction in the total number of layers in each network unit.

2.2 Key Results of the Original Paper

The original paper produced results using a pre trained feature space of 16 convolutions and 5 convergence layers based on a 19-layer VGG network (using averaging rather than max-pooling aggregation for better visual representation). This paper presents an artificial neural network system that enables the separation of image content from style, allowing rewriting of the contents of an image in any other imagery style. The authors have demonstrated this by creating new art images that combine the style of several well-known paintings with the content of arbitrarily selected photographs, that is, images that are derived from a characteristic response of high performance deep NN Representation of Content, Style and Object Recognition.

The images are constructed by using the content representation of the photograph in conjunction with the style representation of the respective piece of art selected for style transfer. While preserving the global layout of the original photograph, the colors and partial structures that make up the global landscape are also provided by the work. In fact, this makes the photo arty so that the composite image looks like a work of art even though it shows the same content as the photo. The most visually appealing images are usually created by matching the style representations to the highest layers in the network.

The algorithms presented by the paper often produce good results in shifting the artistic style of repetitiveness. Unfortunately, otherwise the generated images often do not satisfy the expectations of the people about how the style should be transferred. This is the problem we are trying to solve in this work.

3. Methodology

Taking into consideration the finer nuances from the original paper, we try to look at alternative ways to achieve the required results. We made use of multiple ways of optimization techniques, feature extraction as well as using other networks for achieving the given results.

3.1. Objectives and Technical Challenges

The paper describes the use of the VGG19 network in order to achieve the best performance. As an alternative, we make use of the following two networks:

1. VGG16
2. AlexNet

The primary reason of using the above networks was to try to achieve the results in the original paper with smaller architectures.

We also make use of two ways of optimization techniques:

1. Stochastic Gradient Descent
2. Adam Optimizer

A recent paper [8] showed a comparison of adaptive optimizers such as RMSProp, Adagrad, Adam against SGD. The results depict that performance of the adaptive optimization techniques often are worse at generalization as compared to SGD, even though they have better training performance. We hence consider using a vanilla SGD in comparison with Adam.

The third approach we make use of is using different layers for content and style extraction. As mentioned in the original paper, the conv4_2 layer is used for content extraction, while conv1_1, conv2_1, conv3_1, conv4_1, conv5_1. In this approach, we make use of alternate layers for content extraction as well as style extraction.

3.2. Problem Formulation and Design

The broad description of our methodology used in the concept can be seen as below.

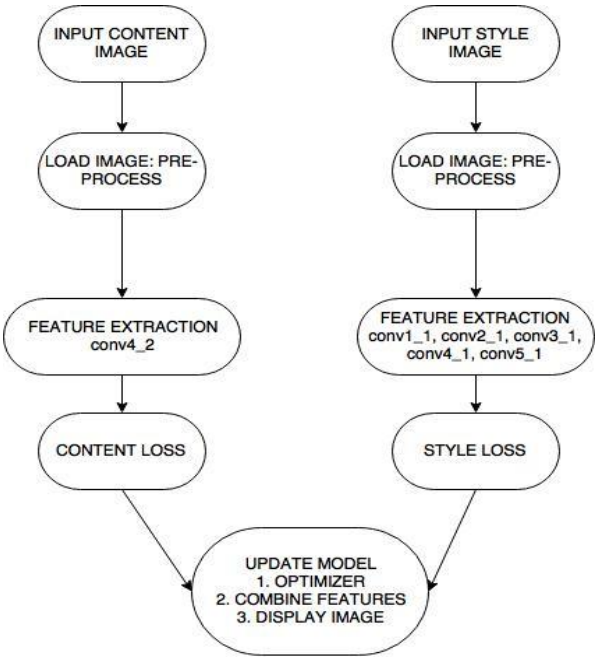

Figure 2: System Overview

4. Implementation

This section talks about deep learning network implemented in this paper in detail along with pseudo code for training algorithms.

4.1. Deep Learning Network

We have implemented VGG16 as well as Alexnet models. Also, we have tried two optimizers stochastic gradient descent (SGD) and Adam.

This section discusses the architectural block diagram of the implementation used in this paper. Then, describes training algorithm in more detail and data used for the same.

4.1.1. Architectural block diagram

Following is the architectural block diagram of the deep learning network implemented in this paper.

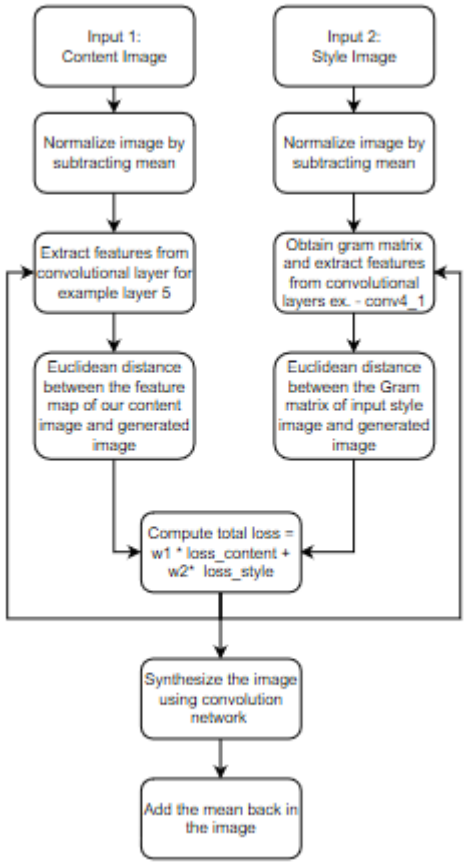

Figure 3: Architectural block diagram

4.1.2. Training Algorithm

This algorithm can be broadly divided in three major parts:

1. Preprocessing the data
2. Extracting the features
3. Constructing output image

Let us discuss each section in detail.

1. There are two input images - one is content image and other one is style image. This paper performs task of transferring the style from style image to the content image and this can be posed as an optimization problem which can be solved through training a neural network.

2. The we preprocess images by subtracting mean.
3. For the 2 images - content and style, outputs from layers are extracted and saved.
   a. VGG Net- Layers used for extracting style from art image are conv1_1, conv2_1, conv3_1, conv4_1 and conv5_1. Layer used for extracting content from photo is conv4_2.
   b. ALEX Net- Layers used for extracting style from art image are Conv Pool layer 1, Conv Pool layer 2, Conv Pool layer 3, Conv Pool layer 4, Conv Pool layer 5. Layer used for extracting content is Conv Pool layer 1.
4. A random image of size 227*227 is generated using the Normal (0,1) distribution.
5. This random image is treated as parameter and the output for this image is recorded from each layer considered for style or content representation.
6. We generated output image.
7. We computed loss function which is weighted sum of two losses.
   a. Content loss $L_{content}$- Given a chosen content layer ι, the content loss is defined as the euclidean distance between the feature map F^l, of our content image C and the feature map $P^l$of our generated image Y. We can write equation as follows:
   $L_{content} = \frac{1}{2}\sum_{i,j} \quad (F_{ij}^l - P_{ij}^l)^2$
   b. Similarly, we can calculate L_{style}. Similarly, given a chosen style layer ι, the style loss is defined as the euclidean distance between the Gram matrix G^l of the feature map of our style image S and the Gram matrix $A^l$of the feature map of our generated image YY. When considering multiple style layers we can simply take the sum of the losses at each layer. We can write the equation as follows:
   $L_{style} = \frac{1}{2}\sum_{i,j} \quad (G_{ij}^l - A_{ij}^l)^2$
   c. We can write final loss function as follows:
   $L = \alpha L_{content} + \beta L_{style}$, where α and β are hyperparameters and we are setting them as a ratio $\frac{\alpha}{\beta}$.

4.1.3. Data Used

For implementation and training the models we used following Artworks from renowned artists, such as The Retrato de Dora Maar painting made by Pablo Picasso, Signs In Yellow by Paul Klee, Rembrandt's Self-Portraits and Starry Night By Van Gogh.

4.2. Software Design

Adam is an optimization algorithm that is used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. Adam has several advantages such as straightforward to implement, Computationally efficient, Little memory requirements, Invariant to diagonal rescale of the gradients, Well suited for problems that are large in terms of data and/or parameters, Appropriate for non-stationary objectives, Appropriate for problems with very noisy/or sparse gradients, Hyper-parameters have intuitive interpretation and typically require little tuning.

Pseudo Code for Adam:

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: f(θ): Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g^2_t$ (Update biased second raw moment estimate)
$m_t' \leftarrow m_t / (1 - \beta^t_1)$ (Compute bias-corrected first moment estimate)
$v_t' \leftarrow v_t / (1 - \beta^t_2)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot m'_t / (\sqrt{v'_t} + \varepsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

SGD is an optimisation technique. It is an alternative to Standard Gradient Descent and other approaches like batch training or BFGS. It still leads to fast convergence, with some advantages such as, doesn't require storing all training data in memory (good for large training sets), allows adding new data in an "online" setting

Pseudo Code for SGD:

procedure SGDTraining(**X, T,W**)
initialize **W** to small random numbers
randomize order of training examples in **X**
while not converged **do**
for n ← 1, N **do**
for k ← 1,K **do**
$y^n_k \leftarrow \sum^d_{i=1} w_{ki} . x^n_i + b_k$
$\delta^n_k \leftarrow y^n_k - t^n_k$

for i ← 1, *d* **do**

$w_{ki} \leftarrow w_{ki} - \eta \cdot \delta^n_k \cdot x^n_i$

end for

$b_k \leftarrow b_k - \eta \cdot \delta^n_k$

end for

end for

end while

end procedure

5. Results

5.1. Project Results

We now take a look at the obtained results, based on both the networks: VGG16 and AlexNet. We observe that the results obtained from the VGG16 network seem to be more clear.

Figure 4: Content Image

Figure 5: VGG16: Style Image, Final Result

Figure 6: AlexNet: Style Image, Final Result

Figure 7: VGG16: Style Image, Final Result

Figure 8: AlexNet: Style Image, Final Result

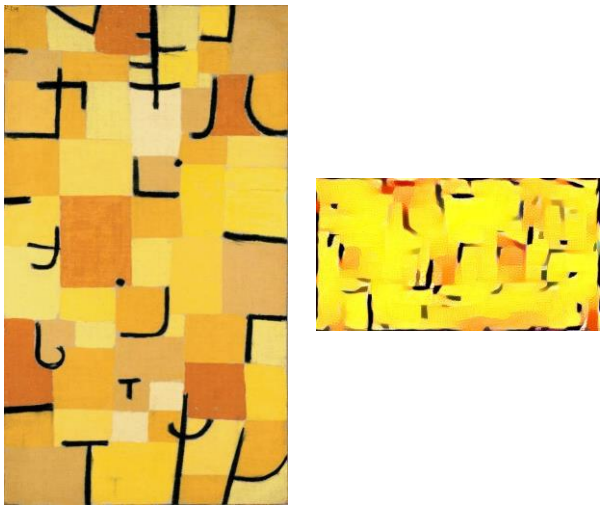

Figure 9: VGG16: Style Image, Final Result

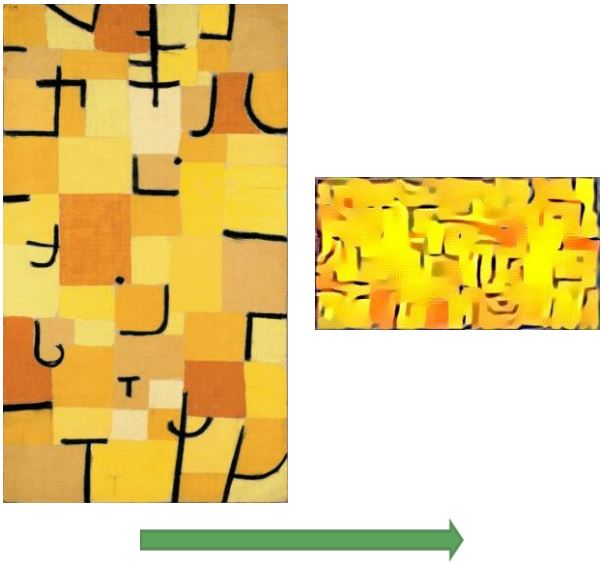

Figure 10: AlexNet: Style Image, Final Result

5.2. Comparison of Results

1. **Comparison with the original paper:** Comparing the results obtained by using VGG16 and AlexNet, we can conclude that we definitely able to achieve similar results by using smaller networks compared to original paper where we are using VGG19. Our models are good since those are shallower than VGG19 and achieve similar output image quality.

2. **Adam vs SGD:**
   Another important factor to compare would be the performance of various optimizers that were used. As discussed before, we make use of the SGD as well as the Adam Optimizer. The figure below shows the L plot for the two optimizers. Though the loss seems to converge comparably well in both approaches(Figure 12 and 13), the output quality in case of Adam is good. The output with Adam optimizer has more fine details (Figure 11) and it resembles more to the original content image compared to SGD. We conclude that using the Adam optimizer seems to be the better approach and it converges faster.

3. **Different α/βratios:**
   As we know in our model we are using α and as weights for $L_{content}$ and L_{style}respectively for computing total loss value and these are hyperparameters and we can play with it.

   In figure 14, we can see the results generated for both $\alpha/\beta = 1e-3$ and $\alpha/\beta = 1e-5$. We can clearly conclude that we achieved good quality of reconstructed images for $\alpha/\beta = 1e-3$. Also, we observed that as we decrease α/βbeyond 1e-3 value we got obscured output images. We could not detect the original content image by looking at our output results.

4. **VGG16 vs Alexnet:**
   As we know that VGG16 is replacement over Alexnet by replacing large kernel-sized filters 11 and 5 in the first and second convolutional layer, respectively with multiple 3X3 kernel-sized filters one after another. Another difference is VGGs are very deep and deeper the network, its performance is good.

   We can say that results of the VGG network are more appealing than that of the ALEX-net. The VGG architecture has significantly more number of layers than the ALEX-net architecture. Thus, making VGG to resemble more with biological visual networks.

Figure 11: Adam vs SGD - Left image is Adam optimizer and right image is for SGD optimizer. Left side image is darker and has fine details.

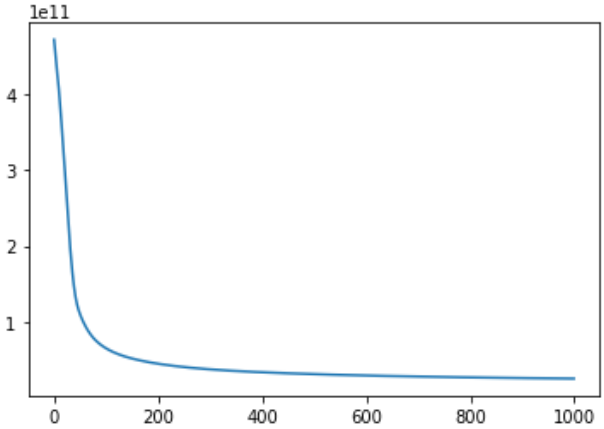

Figure 12: *Loss* plot for Adam optimizer

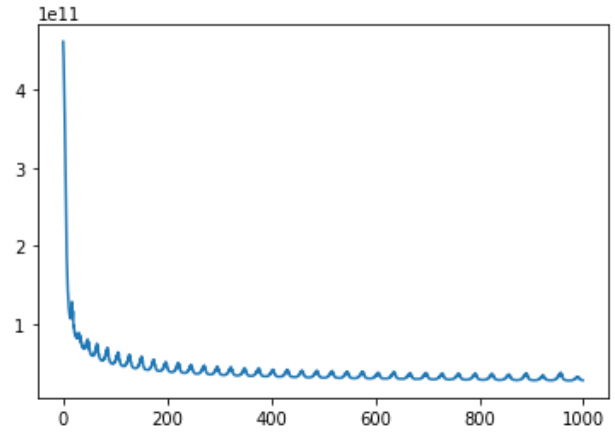

Figure 13: *Loss* plot for SGD optimizer


Figure 14: α/βratio comparison - Left image shows the results for $\alpha/\beta = 1e-3$ and right image shows results for $\alpha/\beta = 1e-5$. We can see clearly right image is blurred.

5.3. Discussion of Insights Gained

One of the most important feature of the original paper, would be the use of the Gram Matrix. The authors suggest that by multiplying the feature extractions by themselves give us a correlation of these features, which is an important aspect, especially while extracting the Style. We also see that the given approach can also be further extended to videos and other graphics as well. As we go deeper into the network these convolutional layers are able to represent much larger scale features and thus have a higher-level representation of the image content.

As far as results are considered, we see that the results obtained from the VGG16 network, seem to be better than AlexNet. An important point here could be the fact that though the original paper uses VGG19 network, we can achieve similar results using much smaller networks. A slight negative can be the fact that since we are merging two images, the overall results are subjective and hence cannot be quantized as art and pictures have subjective appeals.

We make use of 1000 epochs while training the network. However, we can clearly see that L converges much faster and good quality results can hence be easily obtained with far fewer epochs. We observed that the style representation better match as we include higher layers of the network for style feature calculation. This is primarily because the local image structure is better captured when including the contribution of higher layers of the network for calculating style features.

6. Conclusion

1. In our experiments we concluded that VGG16 are better than ALEX-net i.e. deeper layer architectures tend to perform better than shallower networks. But we don't see much difference in VGG16 and VGG19 output images.

2. The representations of style and content in the CNN are separable, and therefore they could be suitably modified to generate perceptually meaningful image.

3. Local image structures captured by the style representation increase in size and complexity when including style features from higher layers of the network. This is primarily because the local image structure is better captured when including the contribution of higher layers of the network for calculating style features.

4. We compared results for Adam vs SGD and observed that Adam is better optimizer since output with Adam has more finer details and resembles more to the original content image.

5. We also compared results for different α/βratios and observed that as value decreases beyond 1e-3 then output image quality becomes poor and image becomes unrecognizable. We got good quality results for α/β= 1e-3.

6. As a further improvement of the project, it would be interesting to have a combination of content and style of arbitrary images, such that it retains the color shades of the content image while learning the style.

7. Another way for improving project is we can use different architectures since VGG has two key drawbacks -

a. It is painfully very slow to train.
b. The network architecture weights themselves are quite large.

We can use smaller architectures like Squeeze net and Google Net.

6. Acknowledgement

This project required in-depth research, a huge amount of work and dedication. Implementation would not have been possible if we did not have the support of many individuals. We would like to extend our sincere gratitude to all of them.

First of all, we are thankful to Professor Zoran Kostic for the provision of expertise and technical support in the completion of this project. Without his knowledge and experience, it would have been impossible for us to complete the project in the given timeline and all the TAs of ECBM4040 (Fall 2017) for their constant guidance and support in coursework, assignment and project.

7. References

[1] https://bitbucket.org/ecbm4040/2017_assignment2_kmm2304/src/69c76d6813b53e14ed431a3ac0a7e760f7569859?at=master

[2] Gatys et al "Image Style Transfer Using Convolutional Neural Networks", CVPR 2016

[3] Gatys et. al., "Texture Synthesis Using Convolutional Neural Networks"

[4] A.Mahendran, A.V edaldi, "Understanding Deep Image Representations by Inverting Them"

[5] *A.Dosovitsky, T.Brox*, "Inverting Visual Representations with Convolutional Networks"

[6] Gatys, Ecker, Bethge , "A Neural Algorithm of Artistic Style"

[7] Gatys, L. A., Ecker, A. S. & Bethge, M. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. arXiv:1505.07376 [cs, q-bio] (2015). URL http://arxiv.org/abs/1505.07376. ArXiv: 1505.07376.

[8]The Marginal Value of Adaptive Gradient Methods in Machine Learning URL https://arxiv.org/abs/1705.08292

8. Appendix

8.1 Individual student contributions - table

	nmp2139	kmm2304	mp3542
Last Name	Paranjape	Mehta	Patel
Fraction of (useful) total contribution	1/3	1/3	1/3
What I did 1	Training of Model with Pablo Picasso Art Work.	Training of Model with Paul Klee Art Work.	Training of Model with Rembrandt's Art Work.
What I did 2	Testing and Hyperparameter optimization with AlexNet, VGG-16.	Testing and Hyperparameter optimization with AlexNet, VGG-16.	Testing and Hyperparameter optimization with AlexNet, VGG-16.
What I did 3	Final Report	Final Report	Final Report

8.2 Architecture of AlexNet

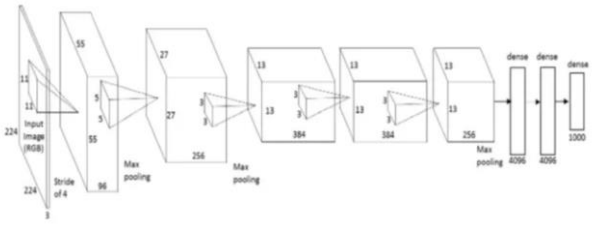

Figure: Architecture of AlexNet

8.3 Architecture of VGG 16

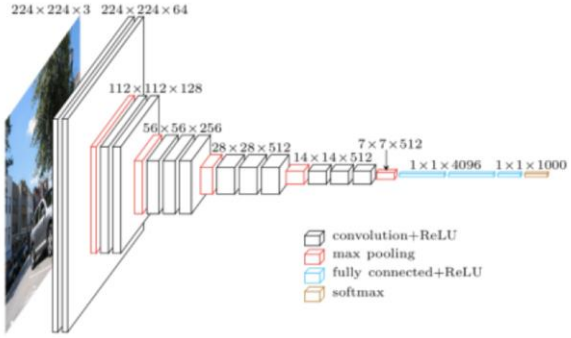

Figure: Architecture of VGG-16